Kathmandu University
Department of Computer Science and Engineering
Dhulikhel, Kavre

Lab Work
“Lab-4”

[Code No: COMP-342]

Submitted by:

Bibhushan Saakha [41]

Submitted to:

Mr Dhiraj Shrestha
Department of Computer Science and Engineering

Submission Date:
2024-06-02

Questions:

Write a Program to implement:

a. 2D Translation
b. 2D Rotation
c. 2D Scaling
d. 2D Reflection
e. 2D Shearing
f. Composite Transformation (Should be able to perform at least 3 transformations)

(For doing these Transformations consider any 2D shapes (Line, Triangle, Rectangle etc), and use Homogeneous coordinate Systems)

Code:

```python
import glfw
from OpenGL.GL import *
from OpenGL.GLU import *
import numpy as np

win_width, win_height = 800, 800

def translate(points, tx, ty):
    translation_matrix = np.array([[1, 0, tx],
                                   [0, 1, ty],
                                   [0, 0, 1]])
    return apply_transformation(points, translation_matrix)

def rotate(points, angle):
    rad = np.radians(angle)
    rotation_matrix = np.array([[np.cos(rad), -np.sin(rad),
0],                             [np.sin(rad), np.cos(rad), 0],
                                [0, 0, 1]])
    return apply_transformation(points, rotation_matrix)

def scale(points, sx, sy):
    scaling_matrix = np.array([[sx, 0, 0],
                               [0, sy, 0],
                               [0, 0, 1]])
    return apply_transformation(points, scaling_matrix)

def shear(points, shx, shy):
    shearing_matrix = np.array([[1, shx, 0],
                                [shy, 1, 0],
                                [0, 0, 1]])
    return apply_transformation(points, shearing_matrix)

def apply_transformation(points, matrix):
    transformed_points = []
    for x, y in points:
        vec = np.array([x, y, 1])
        result = matrix @ vec
        transformed_points.append((result[0], result[1]))
    return transformed_points

def draw_shape(points):
    glBegin(GL_LINE_LOOP)
    for x, y in points:
        glVertex2f(x, y)
    glEnd()

```

```python
def main():
    if not glfw.init():
        return

    window = glfw.create_window(win_width, win_height, "2D Transformations", None, None)
    if not window:
        glfw.terminate()
        return

    glfw.make_context_current(window)

    glViewport(0, 0, win_width, win_height)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    gluOrtho2D(0, win_width, 0, win_height)

    points = [(100, 100), (200, 100), (200, 200), (100, 200)]

    while not glfw.window_should_close(window):
        glClear(GL_COLOR_BUFFER_BIT)

        # Original shape
        draw_shape(points)

        composite_transformation = translate(rotate(translate(points, -100, -100), 30), 100,
100)
        draw_shape(composite_transformation)

        scaled_points = scale(points, 2, 2)
        draw_shape(scaled_points)

        translated_points = translate(points, 400, 200)
        draw_shape(translated_points)

        sheared_points = shear(points, 0.5, 0)
        draw_shape(sheared_points)

        sheared_points = shear(points, 0, 0.5)
        draw_shape(sheared_points)

        # Swap front and back buffers
        glfw.swap_buffers(window)

        # Poll for and process events
        glfw.poll_events()

    glfw.terminate()

if __name__ == "__main__":
    main()

```

This Python program uses GLFW and OpenGL to perform 2D transformations (translation, rotation, scaling, and shearing) on a square shape. It defines functions for each transformation and applies them within a rendering loop to draw the transformed shapes.

Outputs:

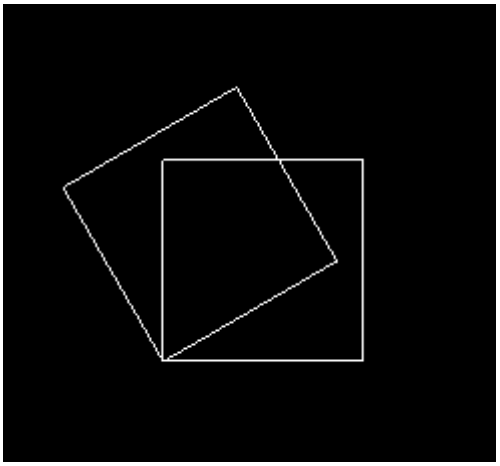

Fig. Composite transformation [translate, rotate, translate back]

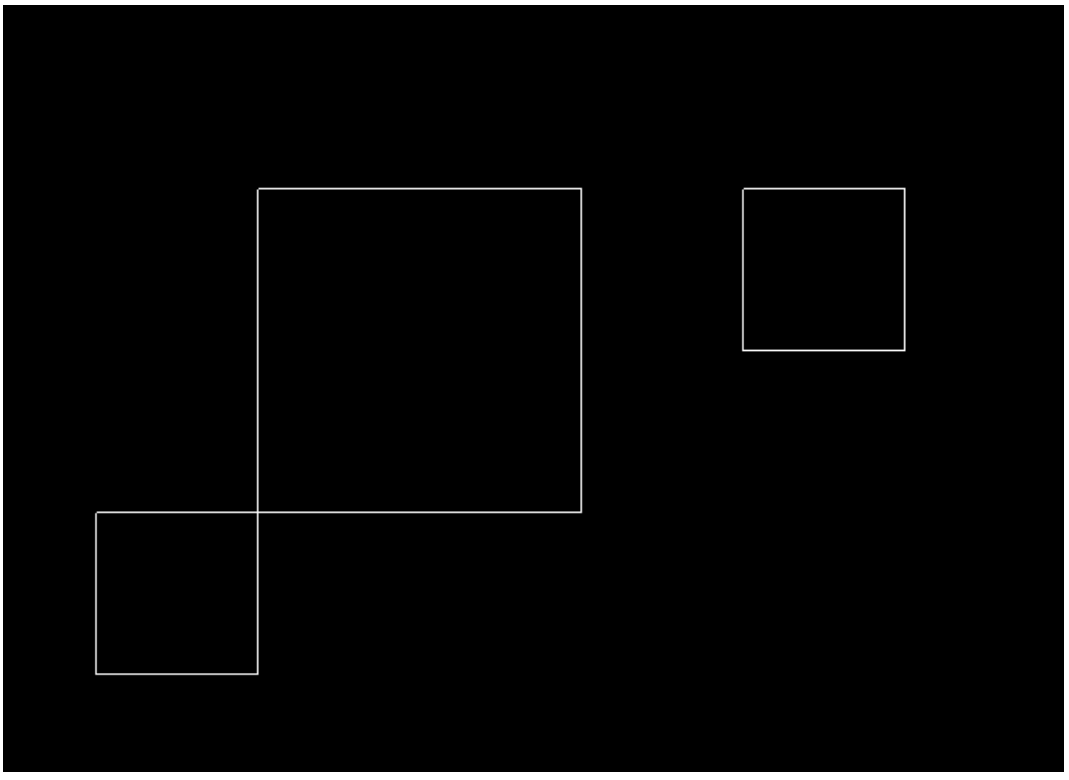

Fig. Scale and Translate

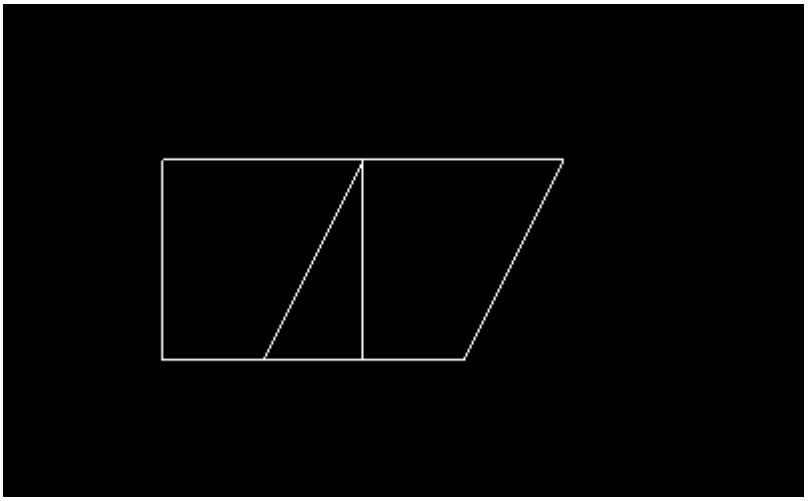

Fig. Shear along x-axis

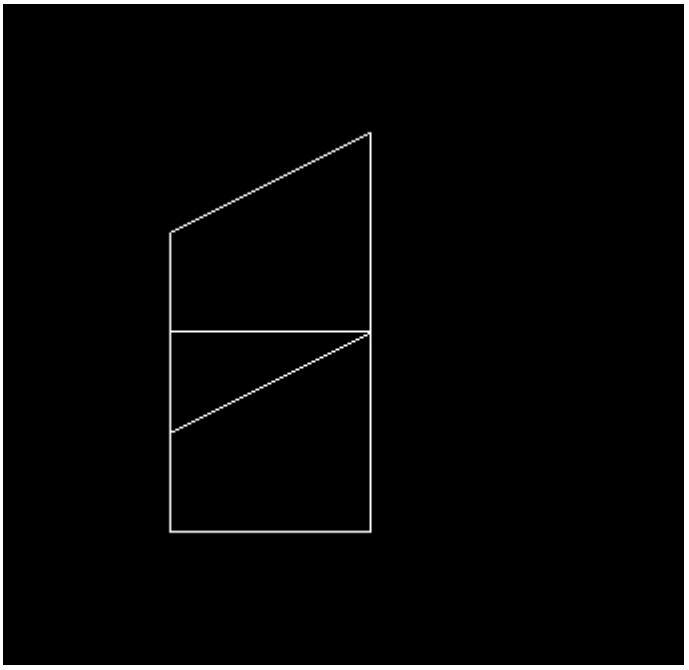

Fig. Shear along y-axis